McMaster University

Software Design Specification

VR Casino: Simulated Virtual Reality for modelling financial risks

Developers: Elvis Chen, Emaad Fazal, Tobi Onireti, Aleem Haq

Table of Contents

1.0: Introduction:

1.1: Purpose:

The purpose of this document is to provide a detailed overview of the project: *Virtual Reality Casino*. The project will be used by a group of researchers to model human behaviour, based on associated financial risk amounts. The simulation will give the researchers the opportunity to observe and record the behavioural patterns of participants who engage in financial risk by playing a slot machine game.

The researchers will choose participants to engage in the virtual reality Casino. They will have full control of the environment based off a few pre-determined algorithms. These algorithms will encompass the list of outcomes that are possible in the game and serve as a basis for the researchers to control the environment.

The VR headset will be used to increase the immersion of the scene. The environment itself will be created using Unity. Input is done through the controllers, allowing the user to choose one of the slot machines in the environment they wish to play.

1.2: Scope:

The following document will contain a full description of the design of the casino and the software.

1.2.1: Unity

The game itself will be designed using Unity. Unity will be used to render the virtual reality environment and all programming involving any sort of user interaction within the environment. For the programming part in particular we will use C# to program any interactions the user has with the environment or animations. C# will be used to generate scripts for the individual objects in the environment.

1.2.2: Blender

For generating the 3D models, we will be using Blender. Blender is a useful 3D modelling tool which allows developers to create immersive and realistic 3D objects in Unity or other game engines. The simplicity of blender and ability to create rigid 3D models is useful for this project as we need to create assets like slot machines for the user to interact with.

1.2.3: C#:

The programming will be done using C#. C# and unity allows users to generate scripts for individual objects which will allow the game environment to behave more as a game rather than a scene. Unity itself is just used to render everything so any sort of logic which is needed will be done in C# by generating scripts for individual objects within the scene and telling C# how to alter the environment.

The results of a trial will be stored in a database using Microsoft SQL SERVER. The researchers need to be able to retrieve past trials for their work so we will use SQL for storing and data retrieval of individual trials.

2.0: Functional Description:

2.1: Machine Selection:

Summary:

The participant has 2 machines for selection. The machines themselves have different betting amounts e.g. machine A has a 10% chance of winning the slot machine game but the amount returned upon a successful win is lower, compared to a machine B, which only has a 1% chance of winning, but with a greater return on winnings.

Sequence:

1.User enters environment and sees 2 slot machines directly in front of them. Each machine will represent either option A or option B from the Holt and Laury: Risk aversion and incentive effects study as shown in the table below:

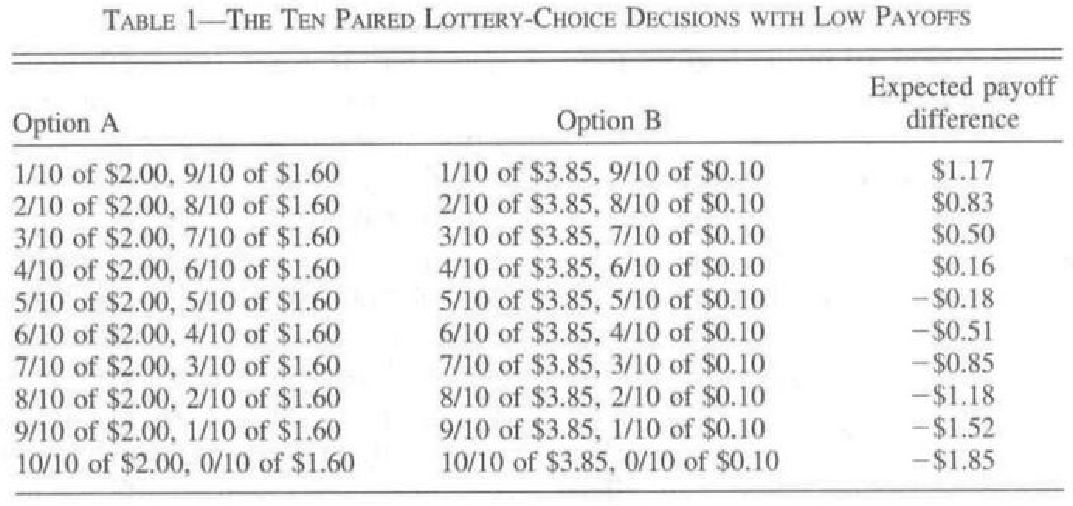


TABLE 1—THE TEN PAIRED LOTTERY-CHOICE DECISIONS WITH LOW PAYOFFS

Option A	Option B	Expected payoff difference
1/10 of \$2.00, 9/10 of \$1.60	1/10 of \$3.85, 9/10 of \$0.10	\$1.17
2/10 of \$2.00, 8/10 of \$1.60	2/10 of \$3.85, 8/10 of \$0.10	\$0.83
3/10 of \$2.00, 7/10 of \$1.60	3/10 of \$3.85, 7/10 of \$0.10	\$0.50
4/10 of \$2.00, 6/10 of \$1.60	4/10 of \$3.85, 6/10 of \$0.10	\$0.16
5/10 of \$2.00, 5/10 of \$1.60	5/10 of \$3.85, 5/10 of \$0.10	−\$0.18
6/10 of \$2.00, 4/10 of \$1.60	6/10 of \$3.85, 4/10 of \$0.10	−\$0.51
7/10 of \$2.00, 3/10 of \$1.60	7/10 of \$3.85, 3/10 of \$0.10	−\$0.85
8/10 of \$2.00, 2/10 of \$1.60	8/10 of \$3.85, 2/10 of \$0.10	−\$1.18
9/10 of \$2.00, 1/10 of \$1.60	9/10 of \$3.85, 1/10 of \$0.10	−\$1.52
10/10 of \$2.00, 0/10 of \$1.60	10/10 of \$3.85, 0/10 of \$0.10	−\$1.85

2. User must select one of two machines and select a few choices of probabilities and win amount(as set by the admin), move in front of their machine of choice, upon which they may now engage in playing the game.

Failure Conditions:

There are no failure conditions in this step as the user must select a machine to continue. If not, then they will remain in the current state of the game. Failure to select a machine does not break the game, it however does leave them in a sort of suspended state since no further progression is possible.

2.2: Slot Game:

Summary:

Upon selecting the machine, the user now sees a user interface for which they can interact with. The user interface itself consists of four slots, a start button, and a field for them to enter how much they wish to bet(or instead show a fixed bet amount, if pre-set by admin). By default, we will start by giving the user one coin, but the amount can be changed by an admin.

Users start the game by pressing the start button or pull the lever and thus randomizing each of the four individual slots with different images. A win occurs when all four images on the individual slots match. The two possible outcomes for the user are win and loss and as mentioned before, if all four images on the slots match, only then does the user win. As a simple example say the user plays and gets four images of the number 7, e.g. **7 7 7 7**, then they win. But, if they get **7 7 7 8** then they lose, as 8 is different than 7. For the simulation itself images will be used instead of numbers.

Sequence of events:

1. User selects a bet amount(if not set by admin), and selects a lottery choice based off the Holt and Laury experiment table.
2. The user presses the start button on the user interface thus commencing the individual trial.
3. The slots are each unique and individually randomized and will stop one by one on a particular image.
4. Feedback is given to user indicating whether they have won or lost.
5. Results of trial and user behaviour are recorded in the database.
6. User is asked if they want to play again.

Failure Conditions:

As before, there are no failure conditions in terms of breaking the game the user can engage in. They will be stuck in the current state if they choose not to press anything. The user cannot advance or play if they choose to not engage in the game. The only failure that can happen in terms of blocking play would be if they choose to bet an amount greater than their available amount. In this case the user is told the amount bet is greater than their available coins. They will be told to bet an amount up to their available money to continue.

3.0: Software System Design:

The following section will be used to describe the environment itself, the slot machine, and the individual slots of the machine. This section will give an overview of the look and design of the machines.

3.1: Environment:

As mentioned previously, the parts of the environment for which the user can interact with encompasses the slot machines themselves and the user interface, which is displayed after selecting an individual slot. The environment itself encompasses more than just those two, however any user interaction will take place with those two parts alone. The non-interactive ‘parts of the environment are:

1. Floor
2. Background

3.2: Floor:

The floor is the part below which the user can see. It is used to add a sense of immersion in the environment and improve the overall experience of the environment. For this case in particular we will design a dark textured floor to add more attraction to the slot machines themselves, which in contrast are more vibrant in color. The dark textured floor is a common pattern seen in regular casinos as it is used to attract users to the lighter colored objects, in our case the slot machine itself. We will add ability for admin to change floor design/colour/texture from some examples.

3.3: Background:

The background like the floor will be dark in texture to create a greater sense of immersion. The darkness in this and the floor make the users view more so towards the objects themselves. We will add ability for admin to change floor design/colour/texture from some examples.

3.4: Slot machine:

The slot machine itself consists of 4 slots and a lever, which the user can pull to start the game. The slot machine is bright in color to attract the user to the machine themselves. This is a common pattern in casinos where the background is kept dark to make users go to the slots. For our case in particular, the slot machine is light in color so when the user enters the environment it will be the first thing they see. We will use blender to add detail to the slot machine itself and add this vibrant color/texture to the slots and add ability for admin to change floor design/colour/texture from some given examples.

3.5: Reels:

Reels are the individual objects within a slot machine which can change in value. By value we mean pictures changing. The reels will be randomly(unless set by admin) generated with images using C#. C# will allow us to create random generators for each reel object which can then be used to display a particular image based on the value generated by C#. For example, say reel A’s random generator returns the number 6 then the image displayed will be whatever we have correlated with the number six. Think along the lines of a hash function where the random number generator generates the number and we put it in the hash function and spit out the corresponding image.

4.0: Backend Logic:

The components of the backend breaks down the internal workings of this project. The components translate to the slot machines and admin configuration. These are broken down into:

- Reel
- Slot Types
- Slot Container
- Admin Portal

4.1: Reels:

This is the smallest unit in the backend. It is the represents the Reel object and it contains the value of the Reel as well as:

- Character of Reel (Icon/Image)
- Constructor is set to a random number
- This can be shown as number value i.e:
    - Value of 1 == Red Heart
    - Value of 2 == 4 Leafed Clover

- Get value method
    - Gets value of the spin result on that reel
- Spin method(no parameter)
    - Sets spin value as random number been 1-n of icons
- Spin method (int Value)
    - Sets spin value as the provided Value
- Spin time method(double Time)
    - Sets the spin time of the reel
    - This is useful for the admin, if they want to change spin time
    - Use default time value if admin doesn't specify

4.2: Slot:

- Constructor is length 4 array of type Reels (each index is a Reels object and probability can be modified by the admin, see slot container below)
    - Runs spin method
- Spin method
- Pay Out method

4.2.1: Win: Slot:

- Spin method
    - Runs reels spin method with no parameter for first reel
    - Run spin(getValue(Reel[0]) for remaining 3 reels
- Pay Out:
    - Give winnings to the user
    - Triggers win feedback

4.2.2: Lose: Slot:

- Spin method - admin set (will have set parameters set for reels entries)
    - Runs reel spin method with no parameter for first reel
    - Runs spin(getValue(Reel[0]) for remaining 2 reels
    - Runs reel spin method with no parameters for last reel
        - The reel.value == Reel[1..3] value
- Possibly add SpinRandomize() - default
- Pay Out:
    - Remove $n amount
    - Triggers loss feedback

4.2.3: Near Misses : Slot:

- Spin method
    - Runs reel spin method with no parameter for first reel
    - Runs spin(getValue(Reel[0] for remaining 2 reels
    - If reel[0].value < position(n) ** if it is not equal to the amount of positions
        - Runs spin(getValue(Reel[0]+1) for last reel
    - Else spin.value = n
        - Runs spin(getValue(Reel[0]-(n-1)) for last reel
    - Pay Out:
        - Remove $n amount
        - Triggers near miss feedback

4.3: Slot Container:

- Array of size n of Type Slots (will be set by the admin)
- Int n
- SlotContainer(int n,double wins, double loses, double nearmisses)
    - This.n to n
    - addwins(wins)
    - addloses(loses)
    - addnms(nearmisses)

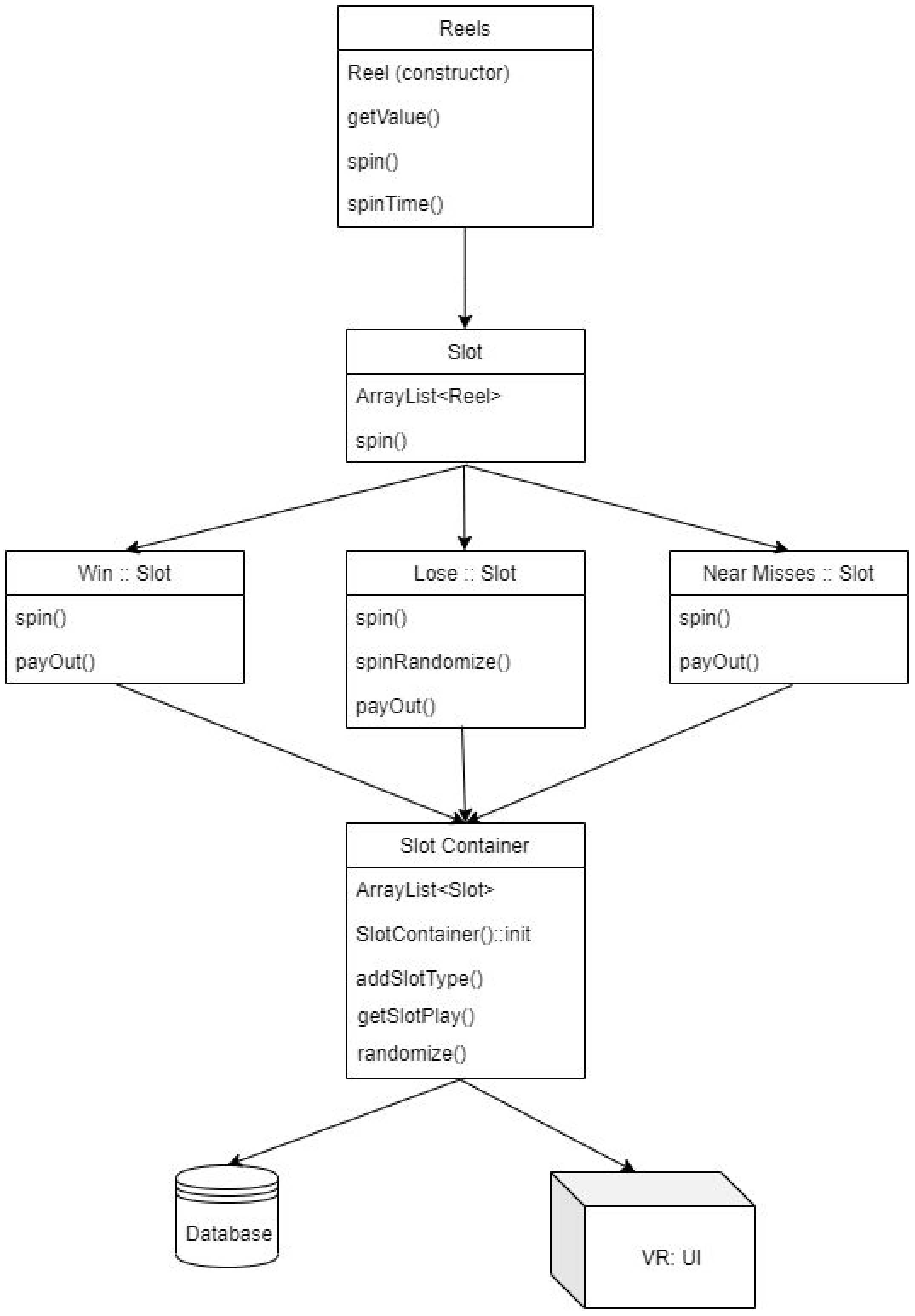
  - Randomizer method for array(randomize the indices)
- addwins(double wins percentage) returns int
  - winresult=(Wins/100)*n
  - Append winresults amount of Wins to SlotContainer array
- addloses(double loses percentage) returns int
  - lossresult=(loses/100)*n
  - Append lossresults amount of Loses to SlotContainer array
  - addnms(double nms percentage) returns int
  - nmresult=(nms/100)*n
  - Append nmresults amount of NearMisses to SlotContainer array
- Randomizer method
  - Randomizes the SlotContainer
- Get Slot Play from SlotContainer(int getPosition)
  - Get the current slot play based off integer input

4.4: Admin Portal:

- Allows the admin to modify different variables such as win probability, payoffs, environment lights and background etc.
- Allows admin to retrieve certain data regarding user sessions, statistics etc.
- Get and Set methods.
- Sets the amount of near misses, loses and wins
- Sets the amount of payout(should be able to input 4 floats, 2 each for Option A and Option B, based off Laury experiment)
- Set the bet amount
- Set the number of starting coins
- Changing background
- Changing floor
- Change slot machine color/texture
- Load defaults if admin doesn't specify
- Get user sessions based off userIds

4.5: Backend Entity Relationship diagram:

An example of the entity relationship for the backend functionality:



5.0: References:

- Risk Aversion and Incentive Effects By Charles A. Holt and Susan K. Laury(2002)